WAIS: A New Development in Information Services

Teresa MacRae, Network Services
Susan Jones, Publication Services

In 1989, Apple Computer, Thinking Machines, and Dow Jones began joint development of WAIS (Wide Area Information Servers), a prototype information resource. Their target market was business executives who wanted quick access to large databases and on-line publications, such as the Wall Street Journal. Although some sources are for-pay and not available to the public, anyone on the Internet can access WAIS.

What Is WAIS?

WAIS automates the search and retrieval of many types of electronic information over wide area networks, such as the Internet. What distinguishes WAIS from other information retrieval systems, such as MIT's TechInfo, is its ability to perform full-text searches as opposed to keyword searches. WAIS searches the full text of all documents in its database for the words specified in a user's request. (Keyword searches match your request only against predefined keywords for each document.)

WAIS provides relatively transparent access to data compared to the two-step telnet/ftp process, where you search an Archie database for topics and then retrieve the data from anonymous FTP sites. With WAIS, once the search is complete, documents automatically display on your machine.

WAIS lets you query a variety of databases on the Internet. It provides a consistent user interface for DOS, Macintosh, and UNIX platforms. The information retrieval protocol acts as a translator between the user interface and many types of database servers. This underlying protocol is an extension of the Z39.50 protocol, which was developed as a standard for library information systems.

What Information Is Available?

Although WAIS is fairly new, there are several for-free databases available. Some examples of for-free databases are:

* The CIA World Factbook
* The Columbia Law School Library Card Catalog
* A poetry server (maintained by MIT)
* Weather maps and forecasts

Future sources may include:
* Articles from the Wall Street Journal and Barron's
* The Library of Congress Card Catalog
* The Tech

How to Get to WAIS

There are several ways to get to WAIS. You can telnet to it, access it from Athena, or use the Macintosh or DOS client versions (see the box below).

Functionality as well as stability varies among the test versions of client software. When you download the Macintosh and DOS versions of WAIS from the anonymous FTP sites, be sure to download the documentation as well.

Resources

At present IS doesn't support WAIS. You can get The WAIS Bibliography, by Barbara Lincoln, via anonymous ftp from <quake.think.com> in the directory /pub/wais/wais-discussion/bibliography.txt.

To receive electronic notices of WAIS progress and releases, send e-mail to <wais-discussion-request@think.com> and ask to be added to their subscription list.

How to Get to WAIS

```
Telnet access: telnet quake.think.com
                login wais
                password your_username

Athena: At the athena% prompt, type add wais; xwais

DOS Windows: anonymous ftp to <ftp.oit.unc.edu> and download
                /pub/wais/UNC/Windows

Macintosh: anonymous ftp to <think.com.> and download
                 /wais/WAIStation-0-63.sit.hqx.
                Note: you will need to use UnStuffIt as well as
                Binhex.
```

A Comparison of Internet Resource Discovery Approaches

Michael F. Schwartz, University of Colorado, Boulder (schwartz@cs.colorado.edu)

Alan Emtage, Bunyip Information Systems (bajan@bunyip.com)

Brewster Kahle, Thinking Machines Corporation (brewster@think.com)

B. Clifford Neuman, USC Information Sciences Institute (bcn@isi.edu)

August, 1992

To appear, *Computing Systems*, 5(4), 1992

Abstract

In the past several years, the number and variety of resources available on the Internet have increased dramatically. With this increase, many new systems have been developed that allow users to search for and access these resources. As these systems begin to interconnect with one another through "information gateways", the conceptual relationships between the systems come into question. Understanding these relationships is important, because they address the degree to which the systems can be made to interoperate seamlessly, without the need for users to learn the details of each system. In this paper we present a taxonomy of approaches to resource discovery. The taxonomy provides insights into the interrelated problems of organizing, browsing, and searching for information. Using this taxonomy, we compare a number of resource discovery systems, and examine several gateways between existing systems.

1. Introduction

For much of the 20 years of its development, the builders of the Internet have been concerned primarily with the improvement of its physical infrastructure. There has been considerable success in this regard, with an increase of four orders of magnitude in the speed and data capacity of the network. Internationally distributed applications that would have been unrealistic to envision even five years ago are now deployed routinely. Examples include wide area distributed file systems, directory services, and group communication systems.

It is estimated that the Internet currently provides direct interactive connectivity to about one million machines world-wide, and periodic (electronic mail/news) connectivity to an additional several hundred thousand machines [Lottor 1992, Quarterman 1992]. This explosive growth has brought with it corresponding growth in the amount of information available to Internet users. We have now reached the stage where many widely accessible information resources are available, including hundreds of gigabytes each of software, documents, sounds, images, and other file system data; library catalog and user directory data; weather, geography, telemetry, and other physical science data; and many other types of information.

Because of this growth in accessible information, the Internet community has begun to show a great deal of interest in the location, retrieval, and management of Internet resources. In the past few years, several user guides have been developed to document the available network information and services [Kehoe 1992, Kochmer 1992, Martin 1991, NSF Network Service Center 1989] that comprise what might be called a burgeoning Internet information infrastructure, or *infostructure*.

Until recently, only a few hundred machines would have been considered "service providers", providing services such as USENET news feeds, "anonymous" FTP[1] archives, WHOIS directory servers, and community specific information, such as bibliographic databases for biological scientists. Knowing who provided each service often required users to consult a local expert, an inefficient use of resources for all parties concerned. Moreover, this approach is impractical in the rapidly changing environment of today's Internet, where any user's machine can offer access to software, documents, and other services.

A number of systems have been developed to provide users access to Internet resources in recent years. These systems come in a variety of forms, and at first may seem to provide unrelated services. The existence and continued construction of gateways to provide interoperation between the systems motivates us to examine the fundamental concepts upon which the systems are built.

In this paper, we examine the interelated issues of organizing, browsing, and searching for information. We present a taxonomy of approaches to these problems, providing insights into the abilities of many of the existing and planned Internet resource discovery services. We begin in Section 2 by discussing the problems of organizing, browsing, and searching. In Section 3 we survey a number of Internet information systems, to provide a base of examples for the taxonomy. We present the taxonomy in Section 4. In Section 5 we use the taxonomy to summarize the design choices made by the systems discussed in Section 3. In Section 6 we summarize the implications of the taxonomy, and conclude with a brief discussion of prospects for the future integration of resource discovery systems.

2. Organizing, Browsing, and Searching

In libraries, highly trained staff are responsible for organizing the available data. Library science has developed methods over hundreds of years to construct a model in which the user, with some experience, can navigate through, locate, retrieve and use the desired information. In contrast, in the Internet every user is also a potential "publisher" and "librarian". No one expects the average user to be able to organize his or her information with such skill. Moreover, because of the decentralized control of Internet information and the difficulty of providing coherent organization in such an environment, most Internet information is only minimally organized. The challenge for the designers of information systems is to help the user find the information that is of interest. Many of the issues here are similar to those that arise in naming research [Bowman, Peterson & Yeatts 1990, Neuman 1992a, Schwartz 1987, Sollins 1985].

[1] FTP is the Internet standard File Transfer Protocol. Anonymous FTP is a convention for allowing Internet users to transfer files to and from machines on which they do not have accounts, for example to support distribution of free software and technical reports.

One method of locating relevant information is *browsing*. By this we mean the user-guided activity of exploring the contents of a resource space. Browsing is closely related to organization, since the better organized the information, the easier it is to browse. Yet by itself, browsing is not sufficient. Because there are few barriers to "publishing" information on the Internet, the Internet contains a great deal of information that is useful to only very few users, and often for only a short period of time. To other users, this information clutters the "information highway", making browsing difficult. Even if all of the information were of interest and well organized, the sheer volume of information can be daunting. For example, in a deeply nested file system with millions of files, browsing to locate a file would be infeasible. In this case, tools are needed that support *searching*. Searching is an automated process, where the user provides some information about the resources being sought, and the system locates some appropriate matches.

Searching in a distributed environment is challenging. Brute force methods such as broadcast can pose a tremendous burden on network resources if the information being sought resides on many machines. In this case, one needs a means by which to limit the scope of searches. One means is to request "advice" from the user about promising places to search. This technique is often helpful, because users may know more about a resource being sought than they initially specify. For example, in trying to locate an electronic mail address, a user may know something about where the person being sought is employed.

If the subject area is sufficiently focused, one might automate this process, by providing what amounts to a rule base of how to search for information in that particular environment. For example, in searching for a particular piece of software, the system might be able to infer that the software runs on top of a particular operating system based on the file name, and narrow the scope of searches to archives containing software for that operating system.

Because of the difficulty of building a rule base or requiring user advice, a common means of supporting searches is to provide an *index* of available information, which can be searched with *flat* search requests (i.e., without regard to how the indexed information is organized). An index can be as simple as a list of file names, or as complex as a relational database with fields corresponding to conceptual characteristics of the information.

The contents of an index has a large impact on how the data can be searched. For example, a search for the string "FTP" in the index of Internet Request For Comments (RFCs) will not yield the result "RFC 959" (which contains the FTP protocol specification), because the title of the document listed in the index ("File Transfer Protocol") does not contain this string.

As this example illustrates, extracting a meaningful characterization of resource data is important. For textual data, brute-force methods such as full-text indexing may be used. Doing so, however, can be space inefficient, and can generate keywords with low meaning (such as the word "and"). Moreover, this meta-data may not provide a sufficient description of the original information. For example, file names are often of little use when trying to determine the contents of a file.

Indexing non-textual data (such as images, sounds, or executable files) presents more difficulties. One promising approach is the use of *transducers*, which extract characterizing information from a file using procedures specific to the type of data contained in that file [Gifford et al. 1991]. For example, subjects may be extracted for mail messages, and procedure names from program source and object files.

As illustrated by the examples above, indices provide a means of interrelating data that is being browsed or searched. The index itself is therefore an example of *meta-data*, or data that organizes the underlying information being sought. Another common means of providing meta-data is a *directory graph*, which is an explicit graph of relationships between objects. For example, the directory tree found in a hierarchical file system is a directory graph. Directory graphs cannot be searched with flat search requests, but rather must be traversed. We will discuss the differences between indices and directory graphs in more depth in Section 4.3.

3. Overview of Resource Discovery Systems

In this section we examine a number of currently deployed Internet information systems, comparing their functionality and approaches to resource discovery. While the systems support different operations and operate in a variety of different domains, there are a number of common aspects of the way they allow users to organize, browse, and search for information. We will explore these aspects in Section 4, using the systems in the current section as a base of examples.

The order in which we discuss these systems is based on a combination of history (to indicate the progression of system development efforts and ideas) and grouping of similar systems together.

3.1. WHOIS

A number of Internet sites run centralized servers that support queries about people and other information across the Internet. One prominent example is the WHOIS service, used by Network Information Centers (NICs) and other organizations to maintain databases of registered users, network numbers, and domains [Harrenstien, Stahl & Feinler 1985]. The user typically specifies the last name of a person being sought, and receives back information including that person's name, work address, telephone number, and electronic mail address. Users can also request site contact information for an Internet domain.

Because each WHOIS server collects geographically distributed information into a single database, it provides a good focal point for registration and searches. However, any one server contains only the small fraction of Internet users and sites that have registered with that NIC, and the information gets out of date because people often forget to inform the NIC when their information changes. Moreover, because each WHOIS server is run independently of the other WHOIS servers (without coordinating content or format), users must explicitly deal with the distribution and inconsistencies between servers. Finally, as the Internet continues to grow, a centralized directory will become a bottleneck and critical point of failure.

3.2. X.500

The Consultative Committee on International Telephony and Telegraphy (CCITT) and the International Organization for Standardization (ISO) have jointly developed a distributed directory service standard called X.500, which describes a hierarchical name space, with provisions for caching, authentication, and replication [CCITT/ISO 1988b]. Each participating site maintains directory information about resources at that site in a Directory System Agent, as well as administrative information needed for traversing the tree and maintaining proper distributed operation. Users access this information through Directory User Agents. There are a number of implementations of X.500 available, and field trials are underway to demonstrate interoperability between the implementations. While X.500 is defined as part of the OSI protocol suite, it can run on top of the Internet through an implementation of the ISO transport service on top of TCP [Rose & Cass 1987].

The most widespread use for X.500 currently is as a user directory. When queried with a fully qualified name of a person (including country, place of employment, etc.), X.500 answers with typed records containing the electronic mail address, telephone number, postal address, and a variety of other information about the person. X.500 can also store other types information. For example, there are projects under way to provide access to various reference documents via X.500. X.500 supports various network services, such as the X.400 electronic mail standard [CCITT/ISO 1988a].

X.500 supports subtree searches. For example, users can browse for the place of employment of a person being sought, and then issue a search request to a server for that part of the tree. It is possible to abbreviate the server search phase to some extent, via a User Friendly Naming mechanism that allows users to provide strings describing the site they want, within a particular country. For example, one can search for the University of Colorado server using the string "colorado", and then search for a person at the University of Colorado with the name of that person.

3.3. archie

A disadvantage of X.500 is that it requires a non-trivial level of effort for a site to install the server software and populate its database with useful information. An increasingly popular way to overcome such problems is to build systems that provide directory service based on existing sources of information, without requiring effort from individual site administrators. This technique is the basis of the archie service, which maintains a list of approximately 1,100 UNIX[2] anonymous FTP archives world-wide, and builds a database of retrievable files by performing recursive directory listings at each site once per month [Emtage & Deutsch 1992]. These sites

[2] UNIX is a registered trademark of UNIX System Laboratories.

currently contain about 150 gigabytes of information, in approximately 2.6 million files. Users can query this database via several interfaces from any of 13 replicated archie servers world-wide, using regular expressions and other types of queries.

Because archie provides an index, searches are not constrained by the hierarchical nature of Internet host names. Users simply specify regular expressions describing the names of files they are trying to locate. In contrast, there is no way for a user to search the X.500 directory service with a similar flat global search. The user must browse the X.500 tree to locate information.

3.4. Prospero

While archie allows users to search for files, the Prospero file system allows users to organize files according to their personal preferences [Neuman 1992b]. In this sense, Prospero is an "enabling technology" for building information infrastructure. Although not a direct source of information itself, Prospero allows users to create their own *views* of the information in a distributed file system. For example, a user might create a view concerning a particular research topic, and populate that view with links to relevant files distributed around the Internet. Other users can then browse this information.

Prospero is based on the Virtual System Model, an approach to organizing large systems that allows users to build their own "virtual systems" from the available resources. A virtual system defines a view of the world centered around the user. Those resources of interest to the user have short names, while the names of objects that the user is less likely to access are much longer. Users can specify parts of their name space as functions of one or more other name spaces. This is accomplished using the *filter*, a user defined program associated with a link, which changes the way one views objects seen through that link; and the *union link*, which makes the contents of a linked subdirectory appear as if they are part of the directory containing the link.

Using Prospero, institutions can maintain directories organizing information in different ways, and these directories can be incorporated into the virtual systems of people who need the information. Among these directories might be indices by author, project, subject, or any other fields. Users can find objects by looking for them in the appropriate index, or by browsing through related virtual systems.

Several global file systems, including the Andrew File System (AFS) [Howard et al. 1988] and the Alex file system [Cate 1992] allow users to form local views of files by creating symbolic links from their own directories. In AFS, the files are restricted to those stored under AFS, while Alex extends the set to files available by anonymous FTP.

3.5. WWW: World Wide Web

Like Prospero, the World Wide Web (WWW) system allows users to organize and access information without concern for the distribution of the information [Berners-Lee et al. 1992]. However, WWW supports two separate discovery models. Part of the information space is based on a hypertext paradigm, where users can explore information by selecting hypertext links to other information. Other parts of the information space consist of indices, which the user encounters while exploring the hypertext space. The user accesses such indices using a flat search paradigm.

3.6. WAIS: Wide Area Information Servers

The Wide Area Information Servers system allows users to deploy, search, and retrieve documents and many other types of information from indexed databases (called "sources") throughout the Internet [Kahle & Medlar 1991]. Information is accessible regardless of format: text, formatted documents, pictures, spreadsheets, graphics, sound, or video.

WAIS was developed by Thinking Machines Corporation, in collaboration with Apple Computer, Inc., Dow Jones & Company, and KPMG Peat Marwick. There are currently over 70 WAIS servers world-wide, offering access to over 300 databases containing technical reports, mailing list and news archives, factual data, classic books and poetry, weather maps, the Bible, and many other types of information. Dow Jones will soon introduce a for-pay server available on their DowVision network, containing several months of the Wall Street Journal and 450 business publications.

While the archie index contains only file names, WAIS indices contain keywords for every word that appears in textual documents (other than common words like "the"). For other kinds of data, WAIS can extract keywords based on knowledge of the particular document type. For example, WAIS understands the structure of a variety of bibliographic database and graphical image formats.

WAIS divides its indices among the servers that provide information, rather than using one global index. A top-level index is provided by a directory of servers operated by Thinking Machines. This index registers information available on each server, including any usage fees.

The decentralized set of WAIS indices have better scalability properties than archie's single index. On the other hand, this decentralization also means that users cannot use flat global searches. Instead, they must first search the directory of servers, and then select particular underlying servers to search.

Users specify searches using natural language queries, such as "tell me about Internet libraries". WAIS does not actually understand the meaning of such queries. Rather, a server responds to a query by applying the words it contains to the full text index of the databases being searched. To obtain the most relevant documents, WAIS ranks matches using a word weighting algorithm. The retrieval process supports a search method called *relevance feedback* [Salton 1986], in which users can request the retrieval of documents based on their similarity (in keyword occurrences) to previously located documents.

3.7. Knowbots™

The Corporation for National Research Initiatives introduced the notion of a Knowbot (Knowledge Robot), which can launch searches for information in a network, possibly replicating itself onto other nodes. Droms implemented an Internet user directory service called the Knowbot Information Service [Droms 1990], which understands the input and output formats of a number of directory services (such as X.500 and WHOIS), and translates user requests as needed to access these services. This technique is similar to the approach used in Schwartz' earlier Heterogeneous Name Service [Schwartz, Zahorjan & Notkin 1987].

3.8. Netfind

Netfind is an Internet user directory service, which attempts to locate electronic mail addresses and other information about Internet users dynamically, using a set of heuristics to locate hosts on which the desired user may have an account or mailbox [Schwartz & Tsirigotis 1991]. The Netfind user specifies the person being sought by first name, last name, or login name, plus one or more keywords describing the name or location of the institution where the user works (e.g., "schwartz university colorado"). The keywords are used to search a *seed database*, to obtain hints of potential administrative domains to search (such as departments within a university or company). This database is gathered by monitoring a number of data sources, including USENET electronic bulletin board messages, WHOIS domain data from several Network Information Centers, logs from various network services, and information supplied by users. Based on the matches from the seed database, the user is asked to select a subset of domains to search. Netfind searches these domains in parallel, as follows. First, each domain is looked up in the Domain Naming System DNS, to locate name servers for the domain. These servers often run on central administrative machines, with accounts and mail forwarding information for many users at a site. Netfind then queries the Simple Mail Transfer Protocol servers on the machines where the name servers run, in an attempt to find mail forwarding information about the specified user. If such information is found, the machines to which mail is forwarded are queried using the *finger* service.

Netfind can often find a user even if the remote site does not support all of these services, or if some steps in the sequence fail. For example, if the finger service is not supported, mail forwarding information may sometimes still be found. Or, if no mail forwarding information is found, Netfind attempts to finger some of the machines listed for that domain in the seed database. This ability to function in the presence of failures or partial remote service support allows Netfind to locate information for over 3 million people in over 5 thousand domains world-wide. Moreover, Netfind can often locate users at sites immediately after they connect to the Internet, because the seed database contains information about many sites beyond those currently connected to the Internet.

3.9. Internet Gopher

The Internet Gopher system provides a simple menu-driven user interface that allows users to browse and locate information from a number of different sources throughout the world [McCahill 1992]. Gopher provides a relatively uniform interface to this data, so that users need not understand many of the details of interacting with each of the systems being accessed. Moreover, Gopher acts as a locus of "registration", providing pointers to many different information sources throughout the Internet. The Gopher user can access information from many of the systems listed in this section, plus various online telephone books, library catalogs, and other data.

4. A Taxonomy of Resource Discovery Systems

Given the diversity of systems described in Section 3, two related questions arise. First, what are the conceptual relationships between these very different looking systems? Second, how can these systems be made to interoperate seamlessly, so that users need not learn the details of each to gain access to the sum of their contents? In the current section we present four characteristics according to which resource discovery systems can be compared. Together, these characteristics form a taxonomy which we use to examine approaches to the resource discovery problems discussed in Section 2, focusing particularly on the systems discussed in Section 3.

The characteristics we introduce concern structural and organizational aspects of abstract "data objects", which are defined by each underlying resource discovery system. For example, files are the data objects in an FTP file system, while data objects could take on values derived dynamically from continuous measurements in an Internet weather service.

Some resource discovery systems distinguish between data (such as files) and meta-data (such as indices or directory graphs). For systems that make such a distinction, the taxonomy can be applied to each level of data. This is useful, because some systems use different implementations for different levels of data. For example, in archie the data are files stored on machines distributed around the Internet, while the meta-data are stored in a centralized index, accessed from RAM using mapped files. In contrast, many FTP sites have "README" files that contain pointers to related archive sites. These pointers are meta-data, yet their implementation is not distinct from the implementation of the other file data.

The characteristics of our taxonomy are *granularity*, *distribution*, *interconnection topology*, and *data integration scheme*. These characteristics can be used to analyze a system for each class of data/meta-data in the system. Thus, our taxonomy has three dimensions. The first dimension consists of the four characteristics. Each characteristic must be considered for each class of data/meta-data supported by a system, forming the second dimension. The systems themselves constitute a third dimension in our analysis.

We examine each of the characteristics below.

4.1. Granularity

The granularity of objects supported by a resource discovery system affects what tasks the system can support. For example, in archie the fundamental resource units are file names, rather than bytes within files, or application-specific divisions, such as individual mail messages or subroutines. Because of this, archie can only be used to locate particular subroutines if they happen to be split into separate files. In a subroutine library that holds many routines in a single file, archie can only be used to locate the overall library file.

This problem could be overcome by using a resource discovery system with a finer-grained indexing mechanism. For example, because WAIS characterizes text files based on their contents rather than just their names, WAIS could be used to index subroutines within files.

The granularity of the index is distinct from the granularity of the data objects. For example, a future version of archie will allow various keywords to be associated with each file, beyond just the file names. Yet, these keywords will still lead only to an overall file (as opposed, for example, to the byte offset within the file at which a particular subroutine starts).

In some systems, resource granularity varies through the resource space. For example, in Netfind the choice of possible domains to search depends on how fine-grained a particular institution choses to divide its computer systems. In some cases, there is only a single domain for an entire site. If the site is large, users may get too many matches in response to their searches. Other sites may divide the domain into very small units. For example, the Computer Science department at Carnegie-Mellon University has nearly 100 subdomains, for individual projects within the department. In this case, individual searches tend to match only a small number of people, but the user must put more effort in to deciding which domains are promising search targets.

Beyond its impact on the user's perception of the information space, data granularity also affects the space overhead of the resource discovery system. In a system that only supports file-level granularity, for example, the index need not store byte offset information. Similarly, a system that generates index keywords based only on file names requires much less space to store the index than a system that generates keywords based on file contents. This difference can be quite large. The ratio of index size to total file data size for archie, for example, is approximately 1:1000, while the corresponding ratio for WAIS is approximately 1:1. If archie used as fine grained an index as WAIS, it would need 150 gigabytes to store the index it currently fits in 150 megabytes. Because individual WAIS indices are stored on different machines, the finer-grained index is feasible.

Gifford's semantic file system [Gifford et al. 1991] supports an index whose granularity varies from object to object, because the transducers extract indexing information from files in different ways, depending on file type.

4.2. Distribution

A spectrum of choices exist for where data and meta-data may be stored. At one extreme, a system could store data in a centralized repository. At the opposite extreme, a system could access data from machines distributed around the world.

A particularly popular design involves a centralized directory for a distributed collection of resources. This design arises in the Internet environment for two reasons. First, providing a resource directory requires administrative effort. Because many Internet sites provide resources (such as FTP files) to the world at no charge, site administrators are often unwilling to put effort into providing a resource directory. This situation favors a design where the resource directory is maintained separately from the resources. Second, while resources are naturally distributed, a centralized directory provides a focal point for searches.

The original archie system is an example of this design. Because this system provided a centralized index of what had previously been available only in a distributed directory graph, archie made it possible to search the data exhaustively, using flat searches. Netfind is a second example of this design. The seed database is a centralized index,[3] allowing users to specify searches using globally flat attributes (namely, the location and institution name where the person being sought resides). In contrast, the user data is extracted from an extremely distributed source — the world-wide collection of computers. This design allows Netfind to locate very timely information about users, in many cases finding where they logged in recently on their personal workstations.

With the advent of replicated servers, the archie index is no longer physically centralized. However, because each archie server attempts to maintain a full replica of the index for all tracked archive sites (rather than dividing the index into disjoint or partially replicated pieces), the current archie network maintains the advantage of a single focal point for searches. Of course, this replication strategy introduces problems with replica consistency, which are the focus of several changes in the next major release (version 3).

While a centralized index allows users to perform flat searches, it can suffer consistency problems as the amount of resource data increases. This problem led archie to settle on a compromise of allowing any piece of directory information to be up to 30 days old. Similarly, the list of domains in the Netfind seed database never perfectly corresponds to the set of all domains in the Internet. In both of these cases, the inconsistency is acceptable because the data in question change relatively slowly. For quickly changing data, a centralized index is difficult to manage.

Rather than maintaining a complete index at each server, another popular design is to use a distributed collection of disjoint directories, with a centralized directory-of-directories. This technique is used by a number of

[3] In the original implementation of Netfind, the seed database was replicated at each site that installed the software. The current Netfind server mechanism allows the index to be centralized and replicated at a small number of sites world-wide.

systems, including WAIS, the Coalition for Networked Information's TopNode project [Percival 1992], Danzig's Indie system [Danzig, Li & Obraczk 1992], and Comer's Directory Location Service [Comer & Norman 1992]. This design arises from the realization that many different resource directories can be created by independent information "curators". Allowing separate underlying directories simplifies administration. In the case of WAIS, the underlying directories are homogeneous: each directory provides access to some number of databases, which can be queried via the WAIS protocol. In the case of the TopNode project, Directory Location Service, and Indie system, the underlying directories are heterogeneous. The information registered for each directory includes an access method or gateway to translate information between formats.

Other than the top-level directory of servers, WAIS stores each directory along with the corresponding resource data. The reasons for co-locating the index and resource data are threefold. First, the indexing mechanism requires access to the entire contents of resource data (as opposed to just the file names, as in archie). Second, the way people use WAIS is to provide an easy way to search through their data, by generating a WAIS index of the data. Therefore, the motivation of decoupling indexing effort from resource provision (as with anonymous FTP files) is not relevant. Finally, because the index and resource data are of comparable size, the usual motivation of providing a small index for a large collection of resource data does not apply.

Like indices and resource data, directory graphs can vary in distribution. X.500 supports a distributed resource directory, where each Directory Service Agent stores a (possibly replicated) piece of the directory tree. As with WAIS, the motivation of decoupling indexing effort from resource provision does not apply to X.500. However, this motivation *does* apply to FTP file systems. For these systems, it is advantageous to decouple the distribution of the directory from the distribution of the resource data. This fact underlies the utility of Prospero. Indeed, perhaps the most popular aspect of Prospero is that it was the first system that supported distributed organization of Internet files. Before Prospero existed, the directory graph had to be on one machine, and the files themselves usually also resided on that one machine. Cross-machine pointers existed only in ad-hoc forms, such as symbolic links or textual descriptions in "README" files.

4.3. Interconnection Topology

To support resource discovery, it must be possible to interrelate resources, so that users may search for and browse them. There are two primary means of doing this. The first technique involves explicit directory graphs, such as those used by X.500, Gopher, Prospero, and the WWW hypertext information space. The second technique involves implicit links in the form of indices, as used by archie, WAIS, Netfind's seed database, and the WWW indices reached by exploring the hypertext space. In these systems the data interconnections are implicit, because objects are related if they share keywords in an index, rather than through a superimposed explicit directory graph.

Interconnection topology affects the ease with which resources can be searched and browsed. X.500 is difficult to search, because the user must know the location in the tree where needed resources reside. However, it is easy to browse the X.500 information space, since it superimposes an explicit hierarchy on the data. In contrast, a particular WAIS database is easy to search, but there is no explicit way to view the relationships that derive from documents sharing keywords (e.g., to see a graph of pointers between related documents).

In general, indices make a system efficient to search, but because they provide only implicit links between related data, there is no way to browse data according to these links. In contrast, directory graphs provide explicit links (and hence support browsing), but provide no means of supporting flat searches. Search efficiency is not an issue in a small centralized environment, where an exhaustive search through the data is feasible. In contrast, if there is a large amount of data or the data are distributed among many machines, exhaustive search is not feasible.

Even in a large, distributed environment, it is possible to selectively search a subset of a resource graph. For example, one can enumerate all possible entries in one or a small number of X.500 servers and compare them with a presented key, even if there is no index. Exactly how many servers are considered feasible to search is typically a matter of administrative control, and user willingness to pay the price (in network charges and delays) for large searches. Because the current Internet does not charge by bandwidth used, users instead limit search scope based on "conventional wisdom" about how large of a search is reasonable. Often these beliefs are based on vague notions of what the technology can support, which change when a new system is introduced. For example, archie showed that it is feasible to collect a large index of widely distributed directory information, and Netfind showed that it is feasible to support dozens of network interactions (such as finger and SMTP connections) per search. Archie and Netfind each changed peoples' attitudes about what types of searches are feasible in a widely distributed environment.

Limiting a search to a "reasonable" number of sites implies the existence of some mechanism to lead the search in promising directions. In the case of X.500, the user specifies which directory servers to search. In the case of Netfind, the user selects a set of domains to search, but then another selection is made to determine which hosts to search at each domain. This latter selection is made by Netfind itself, using a set of heuristics to determine promising hosts within each domain. Such a selection criterion requires that the resource discovery system associate some meaning or type information with the resources it searches. A system that treats resource data as generic, untyped information is in a poorer position to make choices needed to direct the search.

Directory Graphs, Indices, and Hybrid Schemes

In some cases, a directory graph can be built on top of a chain of indices. For example, WAIS uses a two-level indexing scheme, where the directory of servers supports an index that points to individual servers, each with their own indices. While in theory one could select all WAIS sources when performing a search and provide a global flat search capability, the distribution of the indices makes this infeasible. This limitation is identical to the limitation of a directory graph. Essentially, an individual WAIS *server* provides a flat index, but the global WAIS *service* is a hierarchy of WAIS servers. This hierarchy is currently only two levels deep, but it would probably have to grow deeper if, for example, every person in the world wanted to run a WAIS server. This higher level structure for the WAIS service might be provided by other systems, including X.500, Prospero, WWW, or Gopher.

The fact that archie (unlike WAIS) supports a flat global interconnection topology is a consequence of the early state of the current Internet infostructure. As the scale of the global collection of Internet archives grows, a single flat index will no longer be feasible. The next major release of archie (version 3) will incorporate a loose-consistency data distribution mechanism, and split the index into geographical domains. This split is analogous to WAIS sources, but based on geographical location rather than content. Distribution based on content is also planned.[4]

By applying different interconnection topologies for different data levels, hybrid approaches are possible, and in fact are fairly common. For example, at the top level, Prospero supports a directory graph, yet some directories reachable via Prospero (such as the archie database) are flat indices. The result of a query to archie in this case is actually a link back into the graph, allowing one to browse the subdirectory that one finds from an archie query.

Netfind provides another example of a hybrid interconnection topology. At the top level, the seed database provides a centralized index of domains supporting flat global searches. User information is distributed among machines at each of the domains to search, and is interconnected based on the directory graph of the Domain Naming System. This graph is searched using heuristic selection criteria. X.500 also uses a hybrid interconnection topology, but in the opposite order. The list of domains is distributed and (since there is no global index) cannot be searched exhaustively. However, the user information within each domain exists in a flat index that can be exhaustively searched. Because of this difference, a Netfind user can often find many domains but may fail to locate users in those domains, while an X.500 user may be unable to find an appropriate domain, but given an appropriate domain, would be able to locate a user in that domain with certainty (assuming the user is registered in the database). As an aside, these observations indicate that a potential improvement for X.500 would be to provide a global index of domains. Similarly, a flat index would provide better end-person searches for Netfind, although providing indices at each end site will require much more administrative agreement than providing a global domain index.

Attribute-based naming highlights another difference between an index and a directory graph. In attribute-based naming, a user specifies a set of attribute value pairs describing the object to be located [Bowman, Peterson & Yeatts 1990]. To support searches in a graph-based system, the user must specify the order of attributes.

[4] While eventually the archie index may be partitioned into pieces, for the time being each archie server will continue to provide global directory information. However, starting with version 3, the replicas will cooperate in gathering and exchanging this information, so that only one server will retrieve the information from each archive site.

This requirement is made less burdensome in X.500 by providing a canonical order (country, institution, etc.). Nonetheless, in an index-based system that supports queries across indices for each attribute, the order is not important. For example, while X.500 requires that attributes describing a department within a university be placed in a particular order, Netfind's seed database allows these attributes to be specified in any order.

4.4. Data Integration Scheme

An important question for any resource discovery system is how it gains access to data of interest to users. Without a large, evolving collection of data, a resource discovery system will not be used. This consideration has led many resource discovery system builders to focus their prototyping efforts (and in some ways bias their system designs) towards making use of existing Internet infostructure (such as files available by anonymous FTP), and providing gateways to other resource discovery systems.

Populating a system with useful data and providing gateways to other systems raises the question of how to integrate data into the system. This involves two issues: mapping between interconnection topologies, and the mechanics of how and where to perform the mappings.

If the interconnection topologies of two systems are similar (i.e., each uses an index, or each uses a directory graph), mapping between them essentially amounts to mapping between data formats and naming conventions. For example, because AFS and FTP both provide hierarchical file system structures to organize data, Prospero incorporates data from the two systems by simply hiding the specifics of the AFS rooted naming convention (/afs/Internet-domain-name), and by providing a global tree that points to data in separate FTP file systems. If the interconnection topologies of two systems are dissimilar (i.e., one system uses an index and the other uses a directory graph), mapping between them is more difficult. For example, since WAIS provides only implicit connections between resources, it does not readily correspond to the explicit hypertext structure present in the WWW. For such indices, WWW presents the user with a different paradigm, namely flat search. To provide a hypertext view of this data, WWW would essentially need to generate explicit links between each pair of documents that shared common keywords. In addition to the computational expense of doing this, the number of links would be so large that the user would probably get "lost" quickly.

If the data available in two systems are of different granularity, providing mappings between the systems can only effectively be done in one direction. Mapping from a course-grained to a fine-grained system is not possible without either reflecting the lack of information, or using a large amount of external data to supplement the rough-grained information. For example, it would be difficult to populate an X.500 directory with data from Netfind, since Netfind does not provide information about many of the fields that are required by an X.500 directory (such as the title of an individual). Making a gateway from X.500 to Netfind, in contrast, would simply require selecting the needed fields from the X.500 database, and presenting them to the user according to the Netfind display format.

Given the above mapping between interconnection topologies, the next problem is deciding how and where to perform the mapping. There are four basic approaches to making information from one service available through another service: having gateways perform the translation; having the source service support multiple protocols; having the client support multiple protocols; and translating and entering the raw data into the new service.

The first approach, using gateways to perform the translation from one system to another, is used by Gopher and WWW. In these systems, an intermediate server accepts queries from clients using the supported protocol, and translates them into queries understood by the target system. The query is then sent to the target system, and when a response is received it is translated back to a format understood by the client and returned. In WWW, the system keeps track of which gateways support which translations, and forwards the request based on the specified access method. In Gopher, the server with the reference to an external system acts as the gateway.

A disadvantage of gateways is that a gateway can become a bottleneck as an increasing number of users try to use a popular external system. This problem can be remedied by replicating the gateway, but additional steps are needed to balance the load across the replicas. A second problem concerns the use of network bandwidth. In some cases, a small query on one side of a gateway can require the retrieval of a large amount of data on the other side. A related problem is that using a remote gateway to access data physically near the client on the network might result in an extra network round trip. This occurs, for example, when using the WWW-to-WAIS gateway in Switzerland to access a U.S. WAIS server from a U.S. site.

The second approach, that servers support a common protocol, is adopted for meta-data in Prospero. To make meta-data from an existing source available through Prospero, a modified Prospero server is constructed that understands the local data format of the existing service. The Prospero server then exports that data, integrated into the Prospero naming network. Service providers for existing services such as archie, WAIS, or Gopher are then asked to run a Prospero server in addition to the existing server. Updates to the exported information continue using the existing (non-Prospero) methods, and are immediately visible to the Prospero server.

A disadvantage of requiring the source of the information to support multiple protocols is that it is unlikely that every instance of an existing service will be willing to run a new server, making its data available through an additional protocol. For this reason, Prospero also adopts the gateway as a fallback, though such a gateway is considered an interim measure. A second disadvantage is that the Prospero server must be changed when the underlying data format for the existing service changes, whereas with other approaches such a change might fall below the exported interface and therefore be safely ignored.

The third approach is that clients support multiple protocols. This approach is used to access data objects in many systems, including WWW and Prospero. For these systems, the method to be used to access a data object is either explicit in the reference to the object, or can be determined by querying the server on the remote host. The client supports multiple methods to retrieve the object, e.g., FTP, Sun's Network File System, the Andrew File System, or WAIS, and a method supported by the server is used. A disadvantage of this approach is that by supporting multiple protocols the clients become large and less portable. Furthermore, adding a new access method requires an update to all existing clients.

The fourth approach, translating the raw data and entering it into the new service, is used by WAIS. To make the archie database available through WAIS, the filenames from each site in the archie database are listed in a separate document, which is then indexed by WAIS and exported like any other text file. The disadvantage of this approach is that it requires obtaining and processing the entire database from the external service. Depending on the nature of the service, keeping the derived data current may be difficult.

In light of this discussion, we now examine the transformations used between several existing Internet resource discovery systems.

Archie, Prospero, and WAIS

The archie database is available through Prospero and WAIS, using very different transformations. The Prospero-to-archie transformation is performed by a Prospero sever running on each host supporting the archie database. A query is made by listing a Prospero directory that is actually a link into the archie database. This directory name is translated to an archie query, and the list of files matching the query is returned as links in the directory. This transformation allows Prospero to include a great deal of directory information about sites that are not running Prospero servers, but which are tracked by archie.

In contrast, the WAIS-to-archie transformation occurs by treating each archie site listing as a text file, which is then indexed and made available through WAIS.

These different transformations provide very different information. The result of a match in WAIS is a reference to the site listing for sites that matched. If the site listing is then retrieved, it is possible to determine the name of the file on that host, as well as context information (the names of other files in the same directory). The Prospero mapping, on the other hand, does not provide the context information, but instead returns a reference directly to the matched file, not on the archie database host, but on the anonymous FTP site where the file is stored. This eliminates the need to first retrieve the site listing file, which can be quite large.

Gopher, Prospero, and WWW

While no such gateway currently exists, a Prospero server running on a host supporting WWW (or contacting WWW through a gateway) would export WWW documents as directories. In Prospero, even directories can have text associated with them, and this text would be the contents of the document. Each cross reference in the document (called an *anchor*) would be represented as a link to another document from the Prospero directory.

In the other direction, a Prospero directory could be represented as a document whose text is the names of files or subdirectories, and whose anchors correspond to the links in the Prospero directory.

Gopher can be mapped similarly to and from each system. A Gopher menu corresponds to a Prospero directory and a WWW document. The items in the menu correspond to links and anchors.

Other Gopher Gateways

Gopher can make information from almost any service available by connecting the user directly to a client supporting the external service. For example, through Gopher one can access Netfind by logging in to a Netfind client. This style of gateway is useful because it automates the process of connecting to diverse services. It does not, however, perform any translation of the data from the external service, and as such is useful primarily for interactive sessions. Once the user is connected to the external service, a new user interface might have to be learned.

5. System Design Choices

Tables 1 and 2 indicate the design choices made by each of the systems we have discussed in this paper. Because {Systems x Axes x {data, meta-data}} is three dimensional, we have split the table into two pieces, for data and meta data.

System	Granularity	Distribution	Interconnection Topology	Data Integration
archie	Files	Distributed Among Servers World-Wide	N/A	FTP
Gopher	Services, Files	Distributed Among Servers World-Wide	N/A	Gateway
KIS	Users	Distributed Among Several Services	N/A	Multiprotocol
Netfind	Users	Distributed Among End User Machines World-Wide	N/A	Multiprotocol
Prospero	Files	Distributed Among Servers World-Wide	Cross-Reference Graph[4]	Multi-Protocol
WAIS	Documents	Local to each WAIS Server	Relevance Feedback	Data-entered
WWW	Documents	Distributed Among Servers World-Wide	Graph/Anchors[4]	Multi-protocol
WHOIS	Users	Centralized	N/A	Data entered
X.500	General	Distributed Among Servers World-Wide	N/A	Data entered

Table 1: Resource Discovery System Data Design Choices

6. Summary

The first two decades of Internet development were characterized by growth and improvement of the physical network infrastructure. If the trend of the past few years is any indication, the next decade will be characterized by explosive growth in the information infrastructure, or *infostructure*. Already, hundreds of gigabytes each of file system data, library catalog and user directory data, physical science data, and many other types of information are available on the Internet. It stands to reason that the information will grow even faster with the

[4] In Prospero, cross-references are attributes of objects. Therefore, this entry perhaps better appears in the meta-data graph, but cross references in Prospero are more closely tied to objects than the directory information, which provides a higher level of meta-data. In contrast, anchors in WWW are clearly specified in the data objects themselves.

System	Granularity	Distribution	Interconnection Topology	Data Integration
archie	File Names	Centralized and Replicated	Index	Data-entered
Gopher	Varies[5]	Distributed Among Servers World-Wide	Graph	Gateway
KIS	Site Name Keywords	Distributed Among Several Services	Search	Multi-protocol
Netfind	Site Name and Geographical Keywords	Centralized and Replicated	Index and Guided Search	Multi-protocol
Prospero	Varies[5]	Distributed Among Servers World-Wide	Graph	Common Protocol
WAIS	Full Text	Centralized[6]	Index[6]	Data-entered
WWW	Varies[5]	Distributed Among Servers World-Wide	Graph	Gateway
WHOIS	User and Domain Names	Centralized	Index	Data-entered
X.500	Name Components	Distributed Among Servers World-Wide	Graph	Data-entered

Table 2: Resource Discovery System Meta-Data Design Choices

addition of important new constituencies on the Internet, including commercial traffic, K-12 school networking, and digital libraries.

A number of systems have been developed to provide users access to Internet resources in recent years. The existence and continued construction of gateways between these systems raises the important prospect of seamless interoperation between systems. These gateways hint that there may be some fundamental concepts upon which the various systems are built.

In this paper we presented a taxonomy of approaches to the interelated problems of organizing, browsing, and searching for information. The taxonomy's four characteristics (granularity, distribution, interconnection topology, and data integration scheme) represent separate sets of design choices. Yet, looking at the four characteristics together, we see a number of implications that determine the ability to support organizing, browsing, and searching.

The granularity of a resource discovery system impacts the sophistication of searches that can be supported. Coarse-grained systems (such as archie's provision of only file *names*) cannot support as refined queries as systems that provide more fine-grained information (such as WAIS's use of full text indexing). However, finer granularity also implies larger space requirements, which in turn may lead to the need to distribute data. Since WAIS indices are roughly 1,000 times as large as archie indices, these indices have been decentralized from the start in WAIS. As archie incorporates an increasing number of archive sites into its database, it too will begin distributing its index.

Distribution impacts the ease with which data can be accessed, since a centralized system provides an efficient focal point for searches. However, this ease of searching comes at the cost of scalability. When a system contains a large amount of data, services a large user community, or reflects rapidly changing information, it becomes necessary to distribute the data. These motivations led the Internet community to create replica archie servers, and the CCITT/ISO to design distributed data management into X.500.

[5] The granularity of indexing information in Prospero and WWW vary, depending on the user that maintains a directory or document, or the indexing service from which it is derived.

[6] A single WAIS server presents a centralized index. The directory of servers implements a higher-level distributed graph of depth two.

The organization of information in a resource discovery system is based on its interconnection topology. This characteristic affects the ability of the system to support searching and browsing operations. Indexing is at one extreme. This technique supports efficient searches, but provides only implicit interconnections between related data: resources are related only if they happen to share common keywords in the index. Because interconnections are implicit, the user cannot directly view them, and hence cannot browse through the organization of the information. Directory graphs are at the opposite extreme. In this case, the user directly perceives the system organization, and hence can browse the resource space. However, in the absence of an index or a means to limit the scope of a search request, searching a directory graph is inefficient. At best, searches may simply be expensive, as is the case in a recursive descent into a centralized file system hierarchy. At worst, searching without an index can be infeasible, as in the case of searching for a file among the globally distributed collection of FTP archive sites.

There is middle ground in this spectrum. One approach is to limit the scope of index-less searches, based on understanding the semantics or context of the search environment. This approach is taken by Netfind, in selecting a set of hosts to search within a domain. The user may also play a role in narrowing search scope, as is the case when a user selects a set of WAIS sources or a set of Netfind domains to search. Finally, it is possible to support a hybrid interconnection topology, whereby one builds an index on top of or underneath an explicit directory graph. A number of the Internet resource discovery systems are moving in this direction, since doing so can support browsing as well as efficient searching.

Independent of the granularity, distribution, and interconnection topology of a system, there is an important practical issue of how a system gains access to data. Without a large, evolving collection of data, a resource discovery system will not be used. This consideration has led many resource discovery system builders to focus their prototyping efforts (and in some ways bias their system designs) on making use of existing Internet infostructure (such as FTP files), and on providing gateways to other resource discovery systems.

Making use of existing infostructure and providing gateways to other resource discovery systems both rely on a data integration scheme. This involves two issues: mapping between interconnection topologies, and the mechanics of how and where to perform the mappings.

If two systems use similar interconnection topologies (i.e., each uses an index, or each uses a directory graph), mapping between them essentially amounts to mapping between data formats and naming conventions. If the interconnection topologies are dissimilar, mapping between them is difficult or impossible. The easiest solution in this case is simply to present the user with different paradigms (index vs. directory graph), depending on the source of the data.

There are four basic approaches to making information from one service available through another service: having gateways perform the translation; having the source service support multiple protocols; having the client support multiple protocols; and translating and entering the raw data into the new service. No one of these approaches is ideal. A gateway can become a bottleneck, and can waste network bandwidth. Requiring the information source to support multiple protocols makes it unlikely that every instance of an existing service will be willing to run a new server. Requiring that clients support multiple protocols makes clients large and less portable, and makes adding new access methods difficult. Finally, translating and entering raw data presents logistical problems, and in some cases may also lead to consistency problems.

Looking at tables 1 and 2, the natural question to ask is what directions future systems will take to allow the global pool of information to be searched and accessed in a uniform fashion. While it is too early to know what exactly will happen, we see two trends. First, significant efforts are currently under way in the Internet Engineering Task Force to define a universal information identification mechanism. Given such a mechanism, the various systems will more easily be able to access each others' data. Second, widespread deployment of the various systems is starting to uncover some shared experiences. These experiences indicate some generally useful ideas, which can be integrated into each system. For example, supplementing a structured information space with an index has proven to be such a powerful search aid that many systems now incorporate indices. As the set of such facilities present in each system begin to converge, two types of changes will be enabled. First, providing gateways between the systems will be easier, because there will be less need for difficult translations between the systems (e.g., between differing interconnection topologies). Second, the differences between the systems themselves will become less pronounced. At this point systems efforts can combine, providing users with a uniform interface, and a more far reaching information system.

Acknowledgements

Schwartz was supported for this work in part by the National Science Foundation under grants DCR-8420944 and NCR-9105372, a grant from Sun Microsystems' Collaborative Research Program, and a grant from AT&T Bell Laboratories. Neuman was supported for this work in part by the Defense Advance Research Projects Agency under NASA Cooperative Agreement NCC-2-539.

7. Bibliography

[Berners-Lee et al. 1992]
T. Berners-Lee, R. Cailliau, J. Groff and B. Pollermann. World-Wide Web: The Information Universe . *Electronic Networking: Research, Applications and Policy* , 2(1), pp. 52-58, Meckler Publications, Westport CT, Spring 1992.

[Bowman, Peterson & Yeatts 1990]
M. Bowman, L. L. Peterson and A. Yeatts. Univers: An Attribute-Based Name Server. *Software — Practice & Experience*, 20(4), pp. 403-424, Apr. 1990.

[CCITT/ISO 1988a]
CCITT/ISO. *Message Handling: System and Service Overview.* CCITT/ISO, Gloucester, England, Dec. 1988. CCITT Recommendations X.400/ISO IS 10021-1.

[CCITT/ISO 1988b]
CCITT/ISO. *The Directory, Part 1: Overview of Concepts, Models and Services.* CCITT/ISO, Gloucester, England, Dec. 1988. CCITT Draft Recommendation X.500/ISO DIS 9594-1.

[Cate 1992] V. Cate. Alex - A Global Filesystem. *Proc. Usenix File Systems Workshop*, pp. 1-11, Ann Arbor, MI, May 1992.

[Comer & Norman 1992]
D. Comer and V. Norman. Directory Location Service. *Distributed Processing Technical Committee newsletter*, 14(1), pp. 7-12, IEEE, June 1992.

[Danzig, Li & Obraczk 1992]
P. B. Danzig, S. Li and K. Obraczk. Distributed Indexing of Autonomous Internet Services. Tech. Rep., Comput. Sci. Dept., Univ. Southern CA , June 1992.

[Droms 1990]
R. E. Droms. Access to Heterogeneous Directory Services. Proc. 9th Joint Conf. of IEEE Computer and Communications Societies (InfoCom), June 1990.

[Emtage & Deutsch 1992]
A. Emtage and P. Deutsch. Archie - An Electronic Directory Service for the Internet. *Proc. USENIX Winter Conf.*, pp. 93-110, Jan. 1992.

[Gifford et al. 1991]
D. K. Gifford, P. Jouvelot, M. A. Sheldon and J. W. O'Toole, Jr. Semantic File Systems. *Proc. 13th ACM Symp. Operating Syst. Prin.*, pp. 16-25, Oct. 1991.

[Harrenstien, Stahl & Feinler 1985]
K. Harrenstien, M. Stahl and E. Feinler. NICName/Whois. Req. For Com. 954, SRI International, Oct. 1985.

[Howard et al. 1988]
J. Howard, M. Kazar, S. Menees, D. Nichols, M. Satyanarayanan, R. Sidebotham and M. West. Scale and Performance in a Distributed File System. *ACM Trans. Comput. Syst.*, 6(1), pp. 51-81, Feb. 1988. Presented at the 11th ACM Symp. Operating Syst. Prin., Austin, TX, Nov. 1987.

[Kahle & Medlar 1991]
B. Kahle and A. Medlar. An Information System for Corporate Users: Wide Area Information Servers. *ConneXions - The Interoperability Report*, 5(11), pp. 2-9, Interop, Inc., Nov. 1991.

[Kehoe 1992]
B. Kehoe. *Zen and the Art of the Internet: A Beginner's Guide to the Internet.* Dept. Comput. Sci., Widener Univ., Mar. 1992.

[Kochmer 1992]
J. Kochmer. *NorthWestNet User Services Internet Resource Guide.* NorthWestNet, June 1992.

[Lottor 1992]
M. Lottor. Internet Growth (1981-1991). Req. For Com. 1296, Network Information Systems Center, SRI Int., Jan. 1992.

[Martin 1991]
J. Martin. There's Gold in Them Thar Networks! Or Searching for Treasure in all the Wrong Places. Req. For Com. 1290, Ohio State Univ., Dec. 1991.

[McCahill 1992]
M. McCahill. The Internet Gopher: A Distributed Server Information System. *ConneXions - The Interoperability Report*, 6(7), pp. 10-14, Interop, Inc, July 1992.

[NSF Network Service Center 1989]
NSF Network Service Center. *Internet Resources Guide.* NSF Network Service Center, Aug. 1989.

[Neuman 1992a]
B. C. Neuman. The Virtual System Model: A Scalable Approach to Organizing Large Systems. Tech. Rep. 92-06-04, Dept. Comput. Sci. and Engr., Univ. Washington, Seattle, WA, June 1992. Ph.D. Diss.

[Neuman 1992b]
B. C. Neuman. Prospero: A Tool for Organizing Internet Resources. *Electronic Networking: Research, Applications, and Policy*, 2(1), pp. 30-37, Meckler Publications, Westport, CT, Spring 1992.

[Percival 1992]
P. Percival. *TopNode Data Elements.* Dept. Comput. Sci., Indiana Univ., July 1992. Working document describing the fields and other requirements for registration in the TopNode directory.

[Quarterman 1992]
J. S. Quarterman. How Big is the Matrix? *Matrix News*, 2(2), Matrix Information and Directory Services, Austin, TX, mids@tic.com, Feb. 1992.

[Rose & Cass 1987]
M. T. Rose and D. E. Cass. ISO Transport Service on Top of the TCP. Req. For Com. 1006, May 1987.

[Salton 1986]
G. Salton. Another Look at Automatic Text-Retrieval Systems. *Commun. ACM*, 29(7), pp. 648-656, July 1986.

[Schwartz, Zahorjan & Notkin 1987]
M. F. Schwartz, J. Zahorjan and D. Notkin. A Name Service for Evolving, Heterogeneous Systems. *Proc. 11th ACM Symp. Operating Syst. Prin.*, pp. 52-62, Nov. 1987.

[Schwartz 1987]
M. F. Schwartz. Naming in Large, Heterogeneous Systems. Ph.D. Diss., Tech. Rep. 87-08-01, Dept. Comput. Sci. and Engr., Univ. Washington, Seattle, WA, Aug. 1987.

[Schwartz & Tsirigotis 1991]
M. F. Schwartz and P. G. Tsirigotis. Experience with a Semantically Cognizant Internet White Pages Directory Tool. *J. Internetworking: Research and Experience*, 2(1), pp. 23-50, Mar. 1991.

[Sollins 1985]
K. R. Sollins. Distributed Name Management. Ph.D. Diss., MIT/LCS/TR-331, MIT Lab. for Comput. Sci., Feb. 14, 1985.